RESEARCH MACHINES

IEEE — 488 INTERFACE
BASIC USER MANUAL

IEEE-488 Reference Manual

Release 1, November 1980

The policy of Research Machines Limited is one of continuous development and improvement of its products and services, and the right is therefore reserved to revise this document or to make changes in the computer software it describes without notice. RML makes every endeavour to ensure the accuracy of the contents of this document but do not accept liability for any error or omission.

The original labelled distribution disc is regarded as the only proof of purchase and must be produced in order to qualify for an update at a reduced rate. Keep it safe and always work from copies.

Additional copies of this publication may be ordered from Research Machines at the address above. Please ask for "IEEE-488 Reference Manual".

CONTENTS

INTRODUCTION TO THE IEEE-488 BUS

The IEEE-488 standard, which is also known as GP-IB (for General Purpose Interface Bus), specifies a way of interconnecting a number of devices in order for them to exchange data as required by the user. It specifies a connector type, signal line functions, electrical loading and driving characteristics, and protocols necessary for the orderly exchange of information between devices connected by the interface.

BUS STRUCTURE

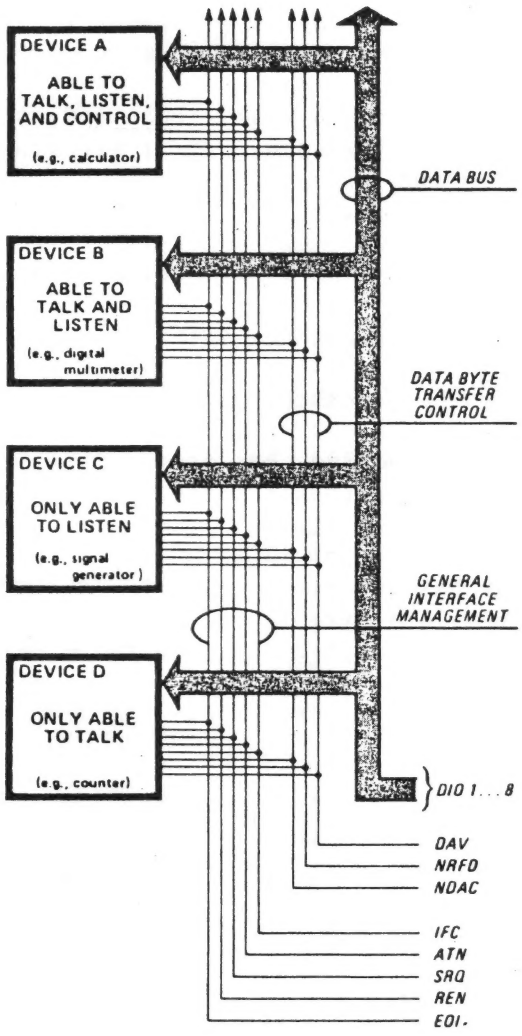


Fig 1
Interface Capabilities and Bus Structure

The GP-IB employs a 16 line bus (together with eight ground lines) to interconnect up to 15 devices. This bus is normally the sole communication link between the interconnected units. Each device on the bus is connected in parallel to the 16 lines of the bus. Eight of the lines are used to transmit data, three are used for communication timing (handshake), and the remaining five for control.

Data is transmitted on the eight GP-IB data lines as a sequence of eight-bit bytes. Data is transferred by means of an interlocked 'handshake' technique. This handshake mechanism allows the bus to operate at the maximum rate of the slowest device addressed at a give time, so that devices with widely differing operating speeds will function correctly when interconnected in one system.

The GP-IB system allows only one device at a time to be an active talker. Up to 14 devices may simultaneously be listeners. Only one device at a time may be an active controller. The GP-IB interface and bus structure are shown in Figure 1.

Communication between devices on the GP-IB employs the three basic functional elements listed below. Every device on the bus must be able to perform at least one of these functions:

LISTENER A device capable of receiving data from other instruments. For example, a printer or a programmable signal generator.

TALKER A device capable of transmitting data to other instruments. For example, a tape reader or a voltmeter outputting data.

CONTROLLER A device capable of managing communications over the GP-IB such as addressing and sending commands. For example, the 380Z.

SIGNAL LINES

The 16 signal lines have the following significance:

DIO1...8 The eight bi-directional data lines.

DAV Data Available. Used by the handshake mechanism.

NRFD Not Ready for Data. Used by the handshake mechanism.

NDAC Not Data Accepted. Used by the handshake mechanism.

ATN Attention. When this line is asserted by the controller in charge the information on the eight data lines is treated as a command by the other devices on the bus.

EOI End or Identify. This line has two purposes. It is used to indicate the end of a multiple byte transfer sequence or, in conjunction with ATN, to execute a parallel polling sequence.

IFC Interface Clear. This is the GP-IB reset line. It can only be asserted by the system controller.

REN Remote enable. The system controller sets REN low and then addresses devices to Listen before they will operate under remote control.

SRQ Service request. This is the GP-IB interrupt line. A device can assert this line to indicate to the controller that it is in need of service.

INSTALLATION

Before installing the board in the 380Z there some user selectable options which should be set. These are detailed in the following sections.

BASE ADDRESS

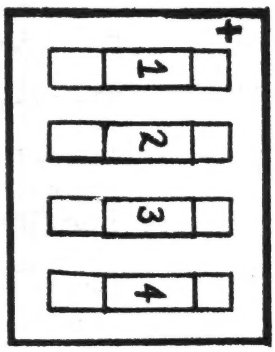
Decide which I/O block the board should occupy. 32 contiguous I/O ports starting on a 32 port boundary are required. As supplied the board is set to use ports 0 to 31. This is the recommended setting and the support software will need modification for use at other addresses. If it is necessary to change the base address, for example to operate two PIO boards in the same machine, proceed as follows.

In the top right hand corner of the board there is a 4 way DIL switch. Switches 1, 2, and 3 set the base address according to table 1. Note that a switch is on when the side marked '+' is depressed.

Switch 1	Switch 2	Switch 3	Base Address
on	on	on	0
on	on	off	32
on	off	on	64
on	off	off	96
off	on	on	128
off	on	off	160
off	off	on	192
off	off	off	224

Table 1

TOP

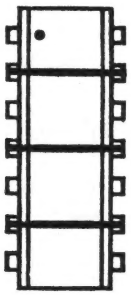


INTERRUPT PRIORITIES

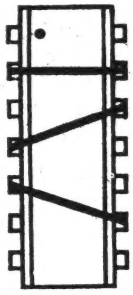
If you do not intend to use the board with interrupts then this section may be skipped. If you do but are unsure of the terms used below, refer to the PIO and CTC technical manuals (available from the Sales Office).

There are four sockets in the top right hand corner of the board marked J5 to J8. J5 (RED) and J6 (BLACK) are the Interrupt Enable In (IEI) and Interrupt Enable Out (IEO) of the PIO/CTC section. J7 and J8 are the corresponding signals for the IEEE section. Linking J5 to J8 (as supplied) gives the IEEE section higher priority than the PIO/CTC section. Linking J6 to J7 gives the PIO/CTC section higher priority than the IEEE section. If this is the only interrupting board, then the remaining two sockets should be left unconnected. If there are several interrupting boards, the remaining RED socket (Board IEI) should be linked to the IEO of the next higher priority board and the remaining BLACK socket (Board IEO) should be linked to the IEI of the next lower board. IEI for the highest board and IEO for the lowest board are left unconnected. For operation with an old PIO board please ask for data sheet 354-240.

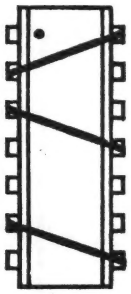
Within the PIO/CTC section the relative priorities of the two PIOs and the CTC are set using links on the header at position EP on the board. As supplied the priorities are, from highest to lowest, CTC, PIO1, PIO2. The diagrams in Figure 2 illustrate the links needed to obtain any set of priorities.



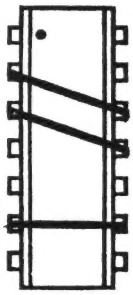
CTC > PIO1 > PIO2



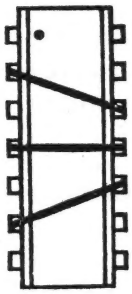
CTC > PIO2 > PIO1



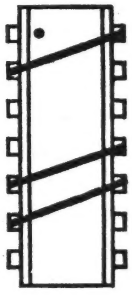
PIO1 > CTC > PIO2



PIO1 > PIO2 > CTC



PIO2 > CTC > PIO1



PIO2 > PIO1 > CTC

Figure 2

CLOCK OUTPUTS

The frequencies of the clock outputs are selected with links on the header at position FP. The clock inputs and data outputs of a dual 4-bit counter (74LS393), the system clock (4MHz), and the Zero Count output of CTC channel 0 are available at the header together with connections to the two clock outputs of the Scotchflex headers. Refer to Figure 3 to make links as required for your application (if any). The example shown in Figure 4 selects a frequency of 125kHz for connector J1 and 250kHz for connector J2. These settings allow the PIO/RTC demonstration software (Version 3) to be used.

Counter 1 Clock In	1	14	Counter 2 Clock In
Counter 1 QA	2	13	Counter 2 QA
Counter 1 QB	3	12	Counter 2 QB
Counter 1 QC	4	11	Counter 2 QC
Counter 1 QD	5	10	Counter 2 QD
System Clock (4MHz)	6	9	CTC Channel 0 Zero Count
Output to J2 pin 33	7	8	Output to J1 pin 33

Figure 3

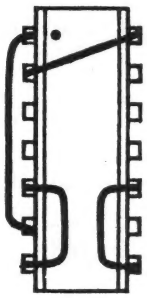


Figure 4

SYSTEM CONTROLLER SWITCH

Switch 4 of the DIL switch in the top right hand corner is connected in parallel with the rear panel toggle switch. Closing either disables the System Controller function of the IEEE interface. The DIL switch can be used to override the rear panel switch and prevent accidental re-enabling if the function will never be required. The board is supplied with the switch open. The System Controller must be enabled to use BASIC with IEEE support, version 1.0.

INSTALLATION OF CONFIGURED BOARD

Install the board in a vacant 380Z slot following the normal procedure for adding or replacing a board. Make sure that the board is the same way round as all of the other boards, and that the IEEE board is positioned under a spare connector on the bus cable. If you are

installing a bus terminator board at the same time as the IEEE board, the terminator board should be positioned at the end of the bus cable away from the CPU board. The 24-way IEEE connector should be mounted on the rear panel by removing the blanking plate from a spare D-type cut-out. Note that on early versions of the IEEE board it is very difficult to insert and remove the IEEE cable. If you do find that you need to remove this cable to mount the connector take great care not to damage the header. Because this header is mounted upside down wire 1 on the IEEE cable (marked with a red stripe on grey cables or coloured brown on rainbow cables) is on the FRONT edge of the cable. If your rear panel does not include a hole suitable for the System Controller switch drill a 1/4" diameter hole adjacent to the connector. Alternatively, if you intend always to use the 380Z as the system controller, the switch can be cut off. If you do this be sure to insulate the ends of the wire.

If you have purchased the PIO internal cables mount the two 37-way D-type connectors by removing two of the longer blanking plates on the rear panel.

When all the cables have been reconnected double check and then turn on the power. If the computer does not behave normally switch off immediately and recheck the connections.

When the IEEE board has been installed and the rest of the computer found to work, you should then verify that the IEEE board works. The BASIC program TESTIEEE.BAS will do this by drawing a square on a Hewlett-Packard 7225A Graph Plotter. This program is listed in Appendix A and, for users with disc systems, supplied on the Distribution Disc.

RML BASICs require that each response to an INPUT statement be terminated with a Carriage Return (CR) character. Because many devices terminate their messages with the EOI message, the INPUT handler inserts CR characters into the input stream if the talker sends EOI. This usually prevents the situation whereby the Talker has finished but BASIC still requires more data.

However, not all devices handle the EOI line in the same manner. The simplest way to deal with such devices is to remove the EOI trap. This is accomplished by the following patch.

Location	Old contents	New contents
EOIPAT	20	00
EOIPAT+1	08	00

The value of the symbolic location EOIPAT is given in the Release Note, which also contains further details about this and any other known problems.

OPERATION FROM BASIC

A special version of BASIC is available in which the INPUT, GET, PRINT, and PUT statements have been extended to allow control of the IEEE-488 bus.

Channel #7 has been reserved for IEEE I/O. Unlike a normal BASIC I/O channel through which data passes to or from only a single device, for example the console or a disc file, the IEEE channel addresses a bus which may have many devices connected to it. Consequently, it is necessary to be able to specify which device or devices on the bus are to take part in any given transaction, and to be able to send commands as well as data along the bus. This is accomplished by including optional command and address information in square brackets immediately following the channel number in the I/O statement. Note that on the VT screen square brackets may be displayed as left and right arrows.

GENERAL DESCRIPTION

The full form of each type of statement is:

```
10 PRINT #7 [addresses/commands], data

20 PUT #7 [addresses/commands], data

30 A=GET(#7 [addresses/commands])

40 INPUT #7 [addresses/commands],variables
```

Note that the PRINT statement will automatically send a carriage-return line-feed after the data unless it is terminated by a semi-colon. This can be avoided by using PUT. However, PUT will convert a numeric expression to a single byte (in the range 0 to 255). The GET function reads a single byte and returns an integer in the range 0 to 255.

A device address on the IEEE bus is a number in the range 0 to 30 (decimal). Hence,

```
10 PRINT #7 [5],A,B
```

addresses IEEE device 5 as a listener and sends the values of A and B as data, whilst:

```
10 INPUT #7 [5],A
```

addresses device 5 as a talker and inputs the value of A. More than one listener may be active at a given time:

```
10 PRINT #7 [5,6],A,B
```

will send the same data to devices 5 and 6.

If the address/command information and the enclosing square brackets are omitted, then the bus is left in the state it was in immediately following the previous #7 operation, and data transfer alone takes place.

For example:

```
10 PRINT #7 [5],A,B
20 PUT #7,X
```

Note that a BASIC expression can be used as an address; it will be evaluated and converted to an integer. However, if the value is outside the range 0 to 30 the error message

```
Illegal function
```

will be printed and execution will stop.

For an INPUT or GET statement the source device (talker) is specified as a single address. Only one talker can be active at once.

```
10 A=GET(#7 [5])
```

reads one byte from device 5.

If more than one address is given, then the extra devices become listeners and receive the same data as the 380Z. This facility is of very limited use but is included for generality. As with output, if no addresses are given then the values used are those of the most recent #7 operation.

It is important to remember that after an output statement the bus is configured with the 380Z as the talker. In order to perform input the bus must be reconfigured with the 380Z as a listener and some other device as the talker. Thus, the first INPUT or GET after an output operation must include a talk address even if this is the same as the previous listen address. Similar considerations apply when performing the first PRINT or PUT after an INPUT or GET. Failure to do this will result in one of the messages:

```
No listeners active
No talker active
```

Commands are given by including a string which is the name of the command (which can be a string expression which evaluates to the name of a command), followed if needed by a number of arguments also within the square brackets and separated by commas. Several separate commands can be given so long as they are separated by commas. Details of the commands are given below. As an example, the statement:

```
10 PRINT #7 [5,"SEC",3,"EOI"],"HELLO";
```

addresses device 5 as a listener, sends the secondary address 3, and then transmits the characters 'H','E','L','L','O' with the EOI message on the last character.

THE COMMANDS IN DETAIL

Certain of the commands can only be used from PRINT or PUT whilst others must be used with GET or INPUT. After each command name in the

description is a list in parentheses of the statements with which that command can be given. The commands are listed in alphabetical order.

ABORT Abort operation (PUT,PRINT)

```
10 PUT #7 ["ABORT"]
```

The ABORT command causes the IEEE Interface Clear (IFC) line to be asserted. All bus activity will be suspended and listeners and talkers will be unaddressed. ABORT should rarely be needed and must be used with care or data may be lost.

CLEAR Clear device (PUT,PRINT)

```
10 PUT #7 ["CLEAR"]    10 PUT #7 [5,6,"CLEAR"]
```

The clear command performs different functions depending on whether any listen addresses are given before the CLEAR. The statement:

```
10 PUT #7 ["CLEAR"]
```

sends the Device Clear (DCL) message to all devices on the bus, whereas:

```
20 PUT #7 [5,6,"CLEAR"]
```

sends the Selective Device Clear (SDC) message only to devices 5 and 6.

CMD Send arbitrary command (GET,INPUT,PUT,PRINT)

```
10 PUT #7 ["CMD",17]
```

It may occasionally be necessary to issue an IEEE command which is not available as one of the special named functions. CMD can be used to send an arbitary command byte on the IEEE bus.

```
10 PUT #7 ["CMD",n]
```

will send n, which must be a numeric expression with a value in the range 0 to 255, as a command byte. In practice CMD will rarely, if ever, be needed.

EOI Send EOI with last byte (PUT,PRINT)

```
10 PRINT #7 [5,"EOI"], "HELLO";
```

This command instructs the interface to send the End or Identify (EOI) message along with the last byte of data to be sent. For example, the statement:

```
10 PRINT #7 [5,"EOI"], "HELLO";
```

will send the EOI message with the character '0'.

INIT Initialize interface (PUT,PRINT)

```
10 PUT #7 ["INIT"]
```

This command must be issued in order to initialise the interface ready for operation. Any program using the IEEE bus should always include:

```
10 PUT #7 ["INIT"]
```

as the first #7 operation or unpredictable results may occur. The INIT command checks the state of the System Controller switch. This switch must be in the 'ENABLE' position or the error message:

```
380Z not system controller
```

will be given.

LOCAL Enable local control (PUT,PRINT)

```
10 PUT #7 [5,"LOCAL"]     10 PUT #7 ["LOCAL"]
```

If the LOCAL command is preceded by listen addresses then a Go to Local (GTL) message is issued to the addressed devices. If no listen addresses have been given then the interface stops sending the Remote Enable (REN) message (if it had previously been sending it). Note that after INIT the REN message is sent automatically.

```
10 PUT #7 [5,"LOCAL"]
```

sends GTL to device 5.

```
10 PUT #7 ["LOCAL"]
```

tells the interface to stop sending REN. The REMOTE command can be used to start sending REN again.

LOCKOUT Lockout local control (PUT,PRINT)

```
10 PUT #7 ["LOCKOUT"]
```

This command sends the Local Lockout (LLO) message. With some programmable instruments LLO can be sent in order to prevent return to local control from the front panel.

MLA My listen address (PUT,PRINT)
MTA My talk address (PUT,PRINT)

```
10 PUT #7 ["MLA",14,"MTA",9]
```

The 380Z has been given listen and talk addresses of 21 (decimal). This choice is arbitrary. In practice it is not necessary to refer to the 380Z's address as it is handled automatically by PUT, PRINT, GET, and INPUT as required. However, if the choice of 21 clashes with the address of another device on the bus it can be changed using the MLA (my listen address) and MTA (my talk address) commands. Thus:

```
10 PUT #7 ["MLA",n,"MTA",m]
```

would set the 380Z to have a listen address of n and a talk address of m. n and m can be the same. Note that INIT will reset the values to 21, so, if a change of address is required the initialization statement should be

```
10 PUT #7 ["INIT","MLA",n,"MTA",m]
```

m and n must be in the range 0 to 30 and not in use by any other device on the bus.

PPOLL Conduct parallel poll (GET)

```
10 A=GET(#7 ["PPOLL"])
```

This command conducts a parallel poll. The value returned is the status byte which can be used to determine whether any devices on the bus are in need of service.

PPOLLC Parallel poll configure (PUT,PRINT)

```
10 PUT #7 [5,"PPOLLC",3,0]
```

This command configures a device for a parallel poll (assuming that the device implements a remotely configurable parallel poll response). The form of the command is:

```
PUT #7 [d,"PPOLLC",n,m]
```

d is the device to be configured. (More than one address can be given but it would not be useful to configure two or more devices to respond in the same way.) The parameters n and m can be any numeric expressions and will be converted to integers. n must be in the range 0 to 7 and specifies on which of the eight data lines the device should respond during a parallel poll. m must be 0 or 1 and defines which state of the selected line will be used by the device to indicate that it requires service. The statement:

```
10 PUT #7 [5,"PPOLLC",3,0]
```

configures device 5 to respond on line 3 during a parallel poll with logical 0 if it requires service.

PPOLLU Parallel poll unconfigure (PUT,PRINT)

```
10 PUT #7 [5,6,7,"PPOLLU"]    10 PUT #7 ["PPOLLU"]
```

This command is used to unconfigure the parallel poll response of devices which have previously been configured with PPOLLC. If no addresses are given, all devices will be unconfigured. If listen addresses are given, only the selected devices will be unconfigured. For example:

```
10 PUT #7 [5,6,7,"PPOLLU"]
```

will unconfigure devices 5, 6, and 7, whereas:

```
10 PUT #7 ["PPOLLU"]
```

will unconfigure all devices.

REMOTE Enable remote control (PUT,PRINT)

```
10 PUT #7 ["REMOTE"]
```

The REMOTE command can be used to instruct the interface to start sending the Remote Enable (REN) message again after it has been stopped by the LOCAL command with no addresses. Some instruments will not respond to their address on the IEEE bus unless the REN message is made true. REN is automatically sent after an INIT command.

SEC Secondary address (GET,INPUT,PUT,PRINT)

```
10 PUT #7 [5,"SEC",3],A
```

The SEC command is used to send secondary device addresses. A secondary address is an integer in the range 0 to 31 (decimal). The command should follow immediately after the associated primary address. For example, the statement:

```
10 PUT #7 [5,"SEC",3],A
```

addresses device 5 as a listener, sends the secondary address 3 and then outputs the value of A. Secondary addresses are often used to select particular functions within a complex device.

SPOLL Conduct serial poll (GET)

```
10 A=GET(#7 ["SPOLL",5])
```

The SPOLL command is used to conduct a serial poll. The

statement:

```
10 A=GET(#7 ["SPOLL",n])
```

conducts a serial poll on device n. The status byte is returned in the variable A. A serial poll will usually be performed after the command SRQ indicates that a device is requiring service.

SRQ Examine service request line (GET)

```
10 A=GET(#7 ["SRQ"])
```

The SRQ command tests the state of the Service Request (interrupt) line of the IEEE bus. When the statement:

```
10 A=GET(#7 ["SRQ"])
```

is executed A will be given the value 0 if no device is requesting service or 1 if one or more devices are requesting service. When a Service Request is detected (A=1) the program can branch to take appropriate action. This will probably be to perform a serial poll to locate the device in need of service.

TRIGGER Group execute trigger (PUT,PRINT)

```
10 PUT #7 ["TRIGGER"]
```

The TRIGGER command sends a Group Execute Trigger (GET) message. The GET message is used to trigger device dependent actions simultaneously in several devices.

THIS PAGE INTENTIONALLY LEFT BLANK

PROGRAMMING

The IEEE section of the board is supported by a special version of BASIC which was described in an earlier section. Demonstration programs for the PIO/CTC section are available in the 'PIO/RTC Software Version 3' available from the Sales Office. This section gives a map of the control registers in the interface together with a brief description of their functions. For full information refer to the data sheets listed in the bibliography.

REGISTER MAP

The addresses given here are relative to the board base address. If the base address is changed from zero, the actual address of the port is found by adding the board base address to the address given.

Port	Note	Register
0		PIO1 Port A Data
1		PIO1 Port B Data
2		PIO1 Port A Control
3		PIO1 Port B Control
4		PIO2 Port A Data
5		PIO2 Port B Data
6		PIO2 Port A Control
7		PIO2 Port B Control
8	1	Buffer control register
9	1	Buffer control register
10	1	Buffer control register
11	1	Buffer control register
12		CTC Channel 0
13		CTC Channel 1
14		CTC Channel 2
15		CTC Channel 3
16		8291 Data
17		8291 Interrupt Mask 1 (write) / Interrupt Status 1 (read)
18		8291 Interrupt Mask 2 (write) / Interrupt Status 2 (read)
19		8291 Serial Poll Mode (write) / Serial Poll Status (read)
20		8291 Address Mode (write) / Address Status (read)
21		8291 Aux Mode (write) / Command Pass Through (read)
22		8291 Address 0/1 (write) / Address 0 (read)
23		8291 8291 EOS (write) / Address 1 (read)
24	2	IEEE Interrupt Monitor Port
25	2	IEEE Interrupt Monitor Port
26		8292 Data
27		8292 Control
28		IEEE Interrupt CTC Channel 0 (8291 Interrupt)
29		IEEE Interrupt CTC Channel 1 (8292 Task completed)
30		IEEE Interrupt CTC Channel 2 (8292 Special Interrupt)
31	3	IEEE Interrupt CTC Channel 3 (8292 Buffer Ready)

NOTES

1. Ports 8,9,10 and 11 are not fully decoded; all four ports access the same register.

2. Ports 24 and 25 are not fully decoded; they both access the same register.

3. Buffer Ready is the logical OR of the 8292 Output Buffer Full and Input Buffer Not Full interrupts.

BUFFER CONTROL PORT

The bits in this output port are assigned as shown below. This port is cleared at power-on or on reset and must be set when the PIOs and CTC are initialized before the interface can be used.

7	6	5	4	3	2	1	0
BYTE/BLOCK	BUF ENAB	BDM2	BDM1	D2B	D2A	D1B	D1A

BYTE/$\overline{\text{BLOCK}}$ This bit is used by the IEEE section. A counter in the 8292 can be used to count bytes transferred if this bit is 1 or blocks transferred if this is 0. A block ends when a byte is transferred with the EOI message.

BUFENAB Setting this bit enables the tri-state buffers on the PIOs and CTC.

BDM2 Setting this bit enables the bi-directional mode for the port 2A buffer.

BDM1 Setting this bit enables the bi-directional mode for the port 1A buffer.

D2B 0 selects Output mode for port 2B.
1 selects Input mode for port 2B.

D2A 0 selects Output mode for port 2A.
1 selects Input mode for port 2A.
This bit is ignored if bit 5 is set.

D1B 0 selects Output mode for port 1B.
1 selects Input mode for port 1B.

D1A 0 selects Output mode for port 1A.
1 selects Input mode for port 1A.
This bit is ignored if bit 4 is set.

INTERRUPT MONITOR PORT

The bits in this input port are assigned as follows:

7	6	5	4	3	2	1	0
NOT USED	NOT USED	IBNFI	OBFI	SPI	TCI	DMA REQ	LTINT

IBNFI Monitors the 8292 Input Buffer Not Full Interrupt pin.

OBFI Monitors the 8292 Output Buffer Full Interrupt pin.

SPI Monitors the 8292 Special Interrupt pin.

TCI Monitors the 8292 Task Completed Interrupt pin.

DMA REQ Monitors the 8291 DMA Request pin.

LTINT Monitors the 8291 Interrupt pin.

The interrupt monitor port is provided to make polled operation easier. In particular, the only way of detecting Task Completed within the 8292 without actually interrupting the Z80 is to enable the 8292 TCI and monitor it in this port.

To operate the IEEE section with mode 2 interrupts the IEEE interrupt CTC should be programmed with each channel set to counter mode, positive edge trigger with a down count of 1. Then, each interrupt request from the 8291 or 8292 will trigger one of the channels and cause a Z80 interrupt.

Note on Reset affecting PIOs

Power-on Reset only supplies a single pulse. However, the PIOs may need 2 pulses to clear external and then internal circuitry in sequence. In certain operating modes of the PIO it may be necessary to press the reset button after switching on the 380Z.

IEEE/PIO/CTC BOARD SPECIFICATION

The board consists of two sections. The first section provides 32 bits of parallel I/O via two Z80A-PIOs and four counter/timers using a Z80A-CTC. The second section, together with suitable software drivers, provides a full implementation of IEEE Standard 488-1978 as given in 'IEEE Standard Digital Interface for Programmable Instrumentation', utilizing the Intel 8291 and 8292 devices.

The board is accessed as a block of 32 contiguous Z80 I/O ports. These ports can be chosen to be on any 32 port boundary using a DIL switch on the board.

PIO/CTC SECTION

32 Data lines in four 8-bit ports.
Each port can be programmed for input or output independently.
2 ports can be used bi-directionally.
8 Handshake lines (two per port).
3 CTC channels with trigger/count input and zero count/time out output.
1 CTC channel with trigger/count input only.

All inputs and outputs are buffered with LSTTL bus tranceivers (74LS244 and 74LS245).

Output logic 0:	<0.4V	12mA max.
Output logic 1:	>2.4V	-3mA max.
Input logic 0:	<0.8V	-0.2mA max.
Input logic 1:	>2.0V	20uA max.

The mode of the buffers, i.e. disabled, input, output or bidirectional is under software control. Note that the bit-mode (mode 3) of the PIOs can only be used by replacing the buffers by DIL headers with appropriate links.

Wait Inputs: 1 LSTTL load with 4k7 pull up to +5V. Logic 0 on either of these inputs asserts the bus wait line.
CAUTION: during a wait state memory is NOT refreshed.

Clock Outputs: The system clock is divided by an 8-stage counter. Outputs from this can be selected by links on a DIL header and made available at the connectors.

Operating temperature: 0-70 deg C

Power requirements: +5V 1.0A typical

Interrupt priority daisy chain look ahead is provided. 2mm jack sockets at the top of the board may be used to extend the chain to other interrupting Z80 peripheral devices. All other connections to this section of the board are made via two 34-way Scotchflex headers at the top of the board. These two connectors have the same pin out; each carries the 16 data lines and four handshake signals from one PIO, trigger inputs and zero count outputs from two CTC channels, a clock output, and a wait input.

PIN CONNECTIONS

J1 - PIO1 and CTC Ch0/1

Board Header 3M type 3431		Rear panel 37-way D-type
Pin	Function	Pin
1	PIO1 Port A strobe	1
2	Common (ground)	20
3	PIO1 Port A ready	2
4	Common	21
5	PIO1 Port B strobe	3
6	Common	22
7	PIO1 Port B ready	4
8	Common	23
9	PIO1 Port A Data 0	5
10	PIO1 Port A Data 1	24
11	PIO1 Port A Data 2	6
12	PIO1 Port A Data 3	25
13	PIO1 Port A Data 4	7
14	PIO1 Port A Data 5	26
15	PIO1 Port A Data 6	8
16	PIO1 Port A Data 7	27
17	Common	9
18	PIO1 Port B Data 0	28
19	PIO1 Port B Data 1	10
20	PIO1 Port B Data 2	29
21	PIO1 Port B Data 3	11
22	PIO1 Port B Data 4	30
23	PIO1 Port B Data 5	12
24	PIO1 Port B Data 6	31
25	PIO1 Port B Data 7	13
26	Common	32
27	CTC Ch0 Zero count	14
28	CTC Ch0 Trigger	33
29	Common	15
30	CTC Ch1 Zero count	34
31	CTC Ch1 Trigger	16
32	Common	35
33	Clock output	17
34	Wait input	36

J2 - PIO2 and CTC Ch2/3

Board Header 3M type 3431		Rear panel 37-way D-type
Pin	Function	Pin
1	PIO2 Port A strobe	1
2	Common (ground)	20
3	PIO2 Port A ready	2
4	Common	21
5	PIO2 Port B strobe	3
6	Common	22
7	PIO2 Port B ready	4
8	Common	23
9	PIO2 Port A Data 0	5
10	PIO2 Port A Data 1	24
11	PIO2 Port A Data 2	6
12	PIO2 Port A Data 3	25
13	PIO2 Port A Data 4	7
14	PIO2 Port A Data 5	26
15	PIO2 Port A Data 6	8
16	PIO2 Port A Data 7	27
17	Common	9
18	PIO2 Port B Data 0	28
19	PIO2 Port B Data 1	10
20	PIO2 Port B Data 2	29
21	PIO2 Port B Data 3	11
22	PIO2 Port B Data 4	30
23	PIO2 Port B Data 5	12
24	PIO2 Port B Data 6	31
25	PIO2 Port B Data 7	13
26	Common	32
27	CTC Ch2 Zero count	14
28	CTC Ch2 Trigger	33
29	Common	15
30	No connection	34
31	CTC Ch3 Trigger	16
32	Common	35
33	Clock output	17
34	Wait input	36

J1 is the 34-way header nearer the rear edge of the board.

J5 (RED) PIO/CTC section Interrupt Enable In

J6 (BLACK) PIO/CTC section Interrupt Enable Out

IEEE SECTION

A CTC is provided as an interrupt controller for this section of the board. This allows the interface to be operated under Mode 2 interrupts. The four channels of the CTC are used as follows:

Channel 0: Byte transferred. A byte has been received over the bus (input mode) or the previous byte has been sent (output mode).

Channel 1: Task completed. A command issued to the controller function has been executed.

Channel 2: Special interrupt. This is given when an event not initiated by the 380Z occurs, for example a Service Request.

Channel 3: Controller buffer ready. This interrupt monitors the data buffer of the controller function. An interrupt is given when the byte written has been accepted, or a byte is waiting to be read.

The system controller function can be disabled with a rear panel switch or with a DIL switch on the board. Closing either switch disables the function. This may be necessary in a system with more than one controller.

The data transfer rate of the interface is limited only by the rate at which the CPU can send bytes to or receive bytes from the interface. In practice this sets an upper limit of about 50,000 bytes/second.

Electrical connection to the board is via a 26-way Scotchflex header. This is taken via a flat cable to the standard 24-way IEEE-488 rear panel connector and the system controller switch.

PIN CONNECTIONS

J4 - IEEE-488 bus

Board Header 3M type 3429		Rear panel 24-way Amphenol
Pin	Function	Pin
1	DIO1	1
2	DIO5	13
3	DIO2	2
4	DIO6	14
5	DIO3	3
6	DIO7	15
7	DIO4	4
8	DIO8	16
9	EOI	5
10	REN	17
11	DAV	6
12	Common	18
13	NRFD	7
14	Common	19
15	NDAC	8
16	Common	20
17	IFC	9
18	Common	21
19	SRQ	10
20	Common	22
21	ATN	11
22	Common	23
23	Common	
24	Common	24
25	Sys Ctrl	
26	Common	

Notes: Pins 25 and 26 on the header connect to the rear panel System Controller switch.

Pin 12 of the Amphenol connector is the chassis ground

J7 (RED) IEEE section Interrupt Enable In

J8 (BLACK) IEEE section Interrupt Enable Out

CABLE REQUIREMENTS

The board occupies one 380Z option slot and one bus connector. If not already installed, a bus cable with at least one spare connector and a bus terminator board will be required. Please refer to the Accessories Price List.

1 off IEEE cable. This is a 26-way flat cable from a 26-way Scotchflex socket to a 24-way Amphenol micro-ribbon connector and System Controller toggle switch. This cable is supplied with the board.

1 off 2mm to 2mm patching lead to set the relative priorities of the IEEE section and PIO/CTC section interrupts. This lead is supplied with the board. An additional patch lead is supplied for use when more than one interrupting board is to be installed in the same machine.

2 off PIO internal cables. These are 34-way flat cables from a 34-way Scotchflex socket to a 37-way 'D' type rear panel connector. These cables will be required if the board is to be connected directly to external equipment. They must be ordered separately. Alternatively, the board may be used in conjunction with a 380Z prototype board within the 380Z case. Under these circumstances the user may have to supply special cables to suit the application. Short 34-way cables terminated in Scotchflex sockets at both ends are available. (Order Code. 797-666).

POWER SUPPLY REQUIREMENTS

+5v @ 900 mA
+12v) Not
-12v) Used

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

A readable introduction to the operation of the GP-IB is contained in:

A Condensed Description of the Hewlett-Packard Interface Bus

It is available from:

Hewlett-Packard Ltd.
King Street Lane,
Winnersh,
Wokingham,
Berkshire.
RG11 5AR

The order number is: 59401-90030

A similar introduction appeared in the June/July issue of Wireless World:

I.E.E.E. bus standard by P.R.Ellefsen

The devices used in this interface are described in full in the Intel data sheets 8291 and 8292.

More details on the controller along with application software examples are given in:

Intel Application Note AP-66
Using the 8292 GP-IB Controller.

The above are available from:

Intel Corporation.
Dorcan House,
Eldene Drive,
Swindon.
SN3 3TU

The final arbiter on any matter concerning the GP-IB is:

IEEE Std. 488-1978
IEEE Standard Digital Interface for Programmable Instrumentation

This document, which is not for the faint-hearted, is available from:

IEEE Service Centre.
445 Hoes Lane,
Piscataway,
New Jersey.
08854
U.S.A.

or through the B.S.I.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

This appendix shows a listing of a simple test program TESTIEEE, supplied on the distribution disc (disc users only) as TESTIEEE.BAS. It draws a square on a Hewlett-Packard 7225A Graph Plotter, assuming that the plotter has a listen address of 5.

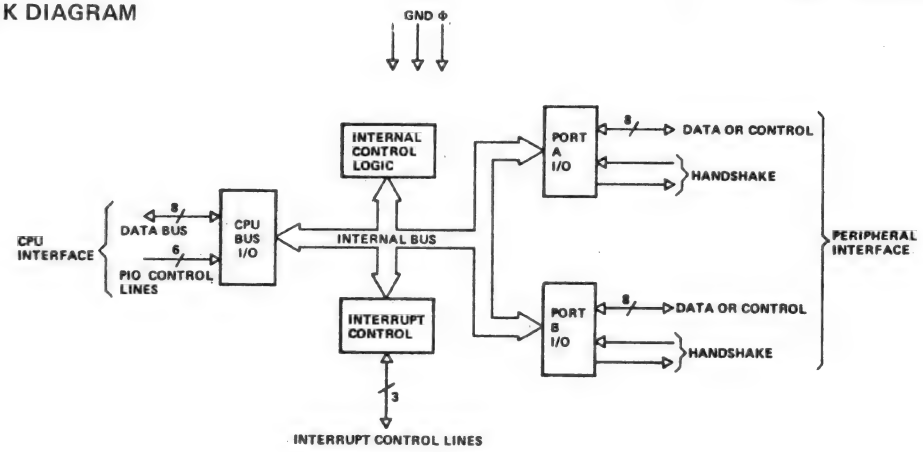
```
10 PUT #7["INIT",5],"IN;" :REM initialize
20 PUT #7,"PU;PA 0,0;PD;" :REM pen up, home, pen down
30 PUT #7,"PA 1000,0;"    :REM 1st side
40 PUT #7,"PA 1000,1000;" :REM 2nd side
50 PUT #7,"PA 0,1000;"    :REM 3rd side
60 PUT #7,"PA 0,0;"       :REM 4th side
70 PUT #7,"PU;"           :REM pen up
80 END
```


APPENDIX B

2.0 PIO ARCHITECHTURE

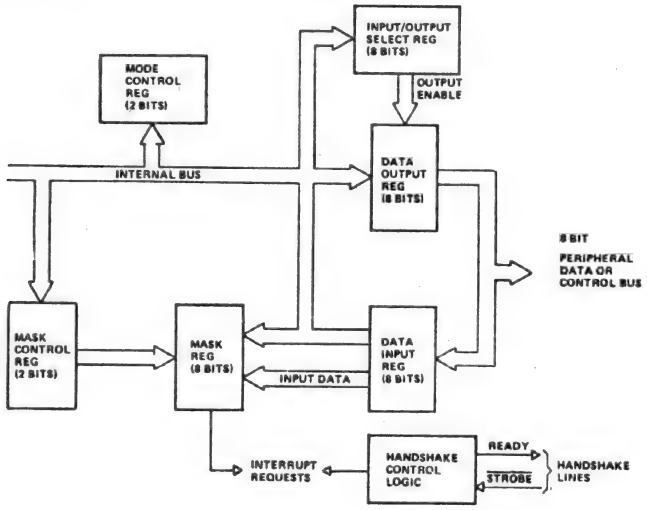
A block diagram of the Z80-PIO is shown in figure 2.0-1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. The CPU bus interface logic allows the PIO to interface directly to the Z80-CPU with no other external logic. However, address decoders and/or line buffers may be required for large systems. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B are virtually identical and are used to interface directly to peripheral devices.

PIO BLOCK DIAGRAM
Figure 2.0-1



The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2.0-2. The registers include: an 8 bit data input register, an 8 bit data output register, a 2 bit mode control register, an 8 bit mask register, an 8 bit input/output select register, and a 2 bit mask control register.

PORT I/O BLOCK DIAGRAM
Figure 2.0-2



The 2-bit mode control register is loaded by the CPU to select the desired operating mode (byte output, byte input, byte bidirectional bus, or bit control mode). All data transfer between the peripheral device and the CPU is achieved through the data input and data output registers. Data may be written into the output register by the CPU or read back to the CPU from the input register at any time. The handshake lines associated with each port are used to control the data transfer between the PIO and the peripheral device.

The 8-bit mask register and the 8-bit input/output select register are used only in the bit control mode. In this mode any of the 8 peripheral data or control bus pins can be programmed to be an input or an output as specified by the select register. The mask register is used in this mode in conjunction with a special interrupt feature. This feature allows an interrupt to be generated when any or all of the unmasked pins reach a specified state (either high or low). The 2-bit mask control register specifies the active state desired (high or low) and if the interrupt should be generated when all unmasked pins are active (AND condition) or when any unmasked pin is active (OR condition). This feature reduces the requirement for CPU status checking of the peripheral by allowing an interrupt to be automatically generated on specific peripheral status conditions. For example, in a system with 3 alarm conditions, an interrupt may be generated if any one occurs or if all three occur.

The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. The priority of any device is determined by its physical location in a daisy chain configuration. Two lines are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output or bidirectional modes, an interrupt can be generated whenever a new byte transfer is requested by the peripheral. In the bit control mode an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routine completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

When an interrupt is accepted by the CPU in mode 2, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector is used to form a pointer to a location in the computer memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant 8 bits of the indirect pointer while the I Register in the CPU provides the most significant 8 bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to a 0 within the PIO since the pointer must point to two adjacent memory locations for a complete 16-bit address.

The PIO decodes the RETI (Return from interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine without any other communication with the CPU.

4.0 PROGRAMMING THE PIO

4.1 RESET

The Z80-PIO automatically enters a reset state when power is applied. The reset state performs the following functions:

1) Both port mask registers are reset to inhibit all port data bits.

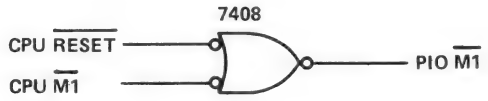
2) Port data bus lines are set to a high impedance state and the Ready "handshake" signals are inactive (low). Mode 1 is automatically selected.

3) The vector address registers are not reset.

4) Both port interrupt enable flip flops are reset.

5) Both port output registers are reset.

In addition to the automatic power on reset, the PIO can be reset by applying an $\overline{M1}$ signal without the presence of a $\overline{RD}$ or $\overline{IORQ}$ signal. If no $\overline{RD}$ or $\overline{IORQ}$ is detected during M1 the PIO will enter the reset state immediately after the $\overline{M1}$ signal goes inactive. The purpose of this reset is to allow a single external gate to generate a reset without a power down sequence. This approach was required due to the 40 pin packaging limitation. It is recommended that in breadboard systems and final systems with a "Reset" push button that a $\overline{M1}$ reset be implemented for the PIO.

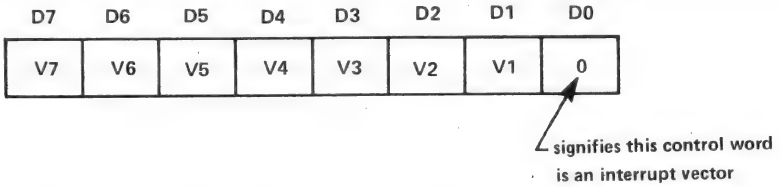


A software RESET is possible as described in Section 4.4, however, use of this method during early system debug may not be desirable because of non-functional system hardware (bus buffers or memory for example).

Once the PIO has entered the internal reset state it is held there until the PIO receives a control word from the CPU.

4.2 LOADING THE INTERRUPT VECTOR

The PIO has been designed to operate with the Z80-CPU using the mode 2 interrupt response. This mode requires that an interrupt vector be supplied by the interrupting device. This vector is used by the CPU to form the address for the interrupt service routine of that port. This vector is placed on the Z80 data bus during an interrupt acknowledge cycle by the highest priority device requesting service at that time. (Refer to the Z80-CPU Technical Manual for details on how an interrupt is serviced by the CPU). The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format:

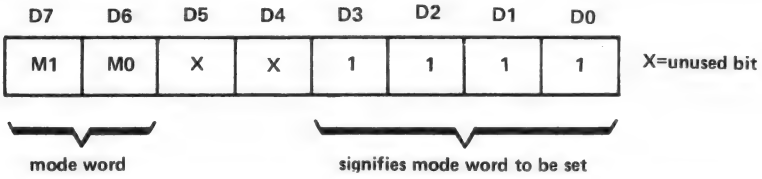


D0 is used in this case as a flag bit which when low causes V7 thru V1 to be loaded into the vector register. At interrupt acknowledge time, the vector of the interrupting port will appear on the Z80 data bus exactly as shown in the format above.

4.3 SELECTING AN OPERATING MODE

Port A of the PIO may be operated in any of four distinct modes: Mode 0 (output mode), Mode 1 (input mode), Mode 2 (bidirectional mode), and Mode 3 (control mode). Note that the mode numbers have been selected for mnemonic significance; i.e. 0=Out, 1=In, 2=Bidirectional. Port B can operate in any of these modes except Mode 2.

The mode of operation must be established by writing a control word to the PIO in the following format:



Bits D7 and D6 from the binary code for the desired mode according to the following table:

D7	D6	MODE
0	0	0 (output)
0	1	1 (input)
1	0	2 (bidirectional)
1	1	3 (control)

Bits D5 and D4 are ignored. Bits D3-D0 must be set to 1111 to indicate "Set Mode".

Selecting Mode 0 enables any data written to the port output register by the CPU to be enabled onto the port data bus. The contents of the output register may be changed at any time by the CPU simply by writing a new data word to the port. Also the current contents of the output register may be read back to the Z80-CPU at any time through the execution of an input instruction.

With Mode 0 active, a data write from the CPU causes the Ready handshake line of that port to go high to notify the peripheral that data is available. This signal remains high until a strobe is received from the peripheral. The rising edge of the strobe generates an interrupt (if it has been enabled) and causes the Ready line to go inactive. This very simple handshake is similar to that used in many peripheral devices.

Selecting Mode 1 puts the port into the input mode. To start handshake operation, the CPU merely performs an input read operation from the port. This activates the Ready line to the peripheral to signify that data should be loaded into the empty input register. The peripheral device then strobes data into the port input register using the strobe line. Again, the rising edge of the strobe causes an interrupt request (if it has been enabled) and deactivates the Ready signal. Data may be strobed into the input register regardless of the state of the Ready signal if care is taken to prevent a data overrun condition.

Mode 2 is a bidirectional data transfer mode which uses all four handshake lines. Therefore only Port A may be used for Mode 2 operation. Mode 2 operation uses the Port A hand-

shake signals for output control and the Port B handshake signals for input control. Thus, both A RDY and B RDY may be active simultaneously. The only operational difference between Mode 0 and the output portion of Mode 2 is that data from the Port A output register is allowed on to the port data bus only when $\overline{A\ STB}$ is active in order to achieve a bidirectional capability.

Mode 3 operation is intended for status and control applications and does not utilize the handshake signals. When Mode 3 is selected, the next control word sent to the PIO must define which of the port data bus lines are to be inputs and which are outputs. The format of the control word is shown below:

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	$IO/_5$	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

If any bit is set to a one, then the corresponding data bus line will be used as an input. Conversely, if the bit is reset, the line will be used as an output.

During Mode 3 operation the strobe signal is ignored and the Ready line is held low. Data may be written to a port or read from a port by the Z80-CPU at any time during Mode 3 operation. (An exception to this is when Port A is in Mode 2 and Port B is in Mode 3). When reading a port, the data returned to the CPU will be composed of input data from port data bus lines assigned as inputs plus port output register data from those lines assigned as outputs.

4.4 SETTING THE INTERRUPT CONTROL WORD

The interrupt control word for each port has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Enable Interrupt	AND/ OR	High/ Low	Masks follows	0	1	1	1
	used in Mode 3 only			signifies interrupt control word			

If bit D7=1 the interrupt enable flip flop of the port is set and the port may generate an interrupt. If bit D7=0 the enable flag is reset and interrupts may not be generated. If an interrupt occurs while D7=0, it will be latched internally by the PIO and passed onto the CPU when PIO Interrupts are Re-Enabled (D7=1). Bits D6, D5 and D4 are used mainly with Mode 3 operation, however, setting bit D4 of the interrupt control word during any mode of operation will cause a pending interrupt to be reset. These three bits are used to allow for interrupt operation in Mode 3 when any group of the I/O lines go to certain defined states. Bit D6 (AND/OR) defines the logical operation to be performed in port monitoring. If bit D6=1, and AND function is specified and if D6=0, an OR function is specified. For example, if the AND function is specified, all bits must go to a specified state before an interrupt will be generated while the OR function will generate an interrupt if any specified bit goes to the active state.

Bit D5 defines the active polarity of the port data bus line to be monitored. If bit D5=1 the port data lines are monitored for a high state while if D5=0 they will be monitored for a low state.

If bit D4=1 the next control word sent to the PIO must define a mask as follows:

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is zero will be monitored for generating an interrupt.

The interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

If an external Asynchronous interrupt could occur while the processor is writing the disable word to the PIO (03H) then a system problem may occur. If interrupts are enabled in the processor it is possible that the Asynchronous interrupt will occur while the processor is writing the disable word to the PIO. The PIO will generate an INT and the CPU will acknowledge it, however, by this time, the PIO will have received the disable word and de-activated its interrupt structure. The result is that the PIO will not send in its interrupt vector during the interrupt acknowledge cycle because it is disabled and the CPU will fetch an erroneous vector resulting in a program fault. The cure for this problem is to disable interrupts within the CPU with the DI instruction just before the PIO is disabled and then re-enable interrupts with the EI instruction. This action causes the CPU to ignore any faulty interrupts produced by the PIO while it is being disabled. The code sequence would be:

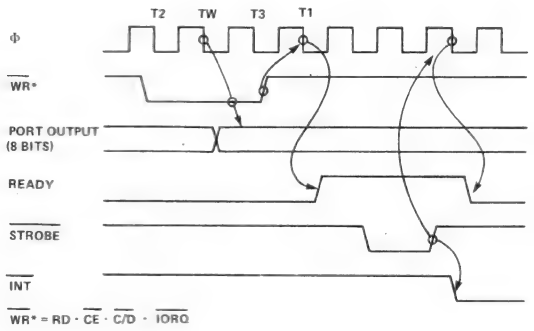
```
.
.
LD  A,03H
DI              ; DISABLE CPU
OUT (PIO),A     ; DISABLE PIO
EI              ; ENABLE CPU
.
.
```


5.0 TIMING

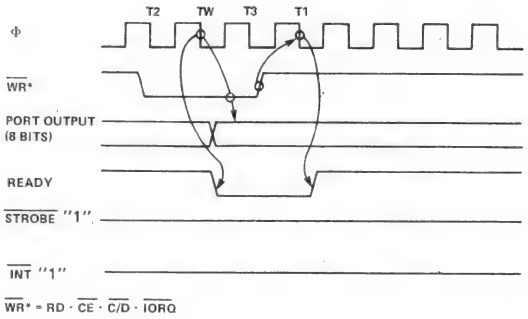
5.1 OUTPUT MODE (MODE 0)

Figure 5.0-1a illustrates the timing associated with Mode 0 operation. An output cycle is always started by the execution of an output instruction by the CPU. A $\overline{WR}$* pulse is generated by the PIO during a CPU I/O write operation and is used to latch the data from the CPU data bus into addressed port's (A or B) output register. The rising edge of the $\overline{WR}$* pulse then raises the READY line after the next falling edge of Φ to indicate that data is available for the peripheral device. In most systems, the rising edge of the READY signal can be used as a latching signal in the peripheral device. The READY signal will remain active until a positive edge is received from the $\overline{STROBE}$ line indicating that the peripheral has taken the data shown in Figure 5.0-1a. If already active, READY will be forced low 1½ Φ cycles after the falling edge of $\overline{IORQ}$ if the port's output register is written into. READY will return high on the first falling edge of Φ after the rising edge of $\overline{IORQ}$ as shown in figure 5.0-1b. This action guarantees that READY is low while port data is changing and that a positive edge is generated on READY whenever an Output instruction is executed.

MODE 0 (OUTPUT)TIMING
Figure 5.0-1a



MODE 0 (OUTPUT) TIMING
Figure 5.0-1b

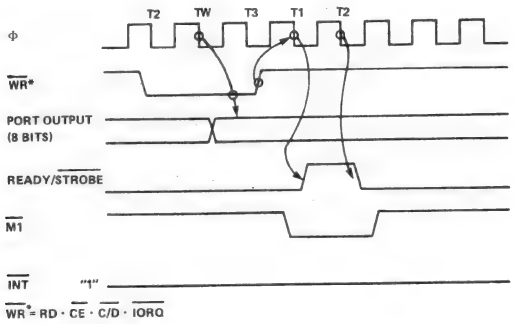


By connecting READY to $\overline{STROBE}$ a positive pulse with a duration of one clock period can be created as shown in Figure 5.0-1c. The positive edge of READY/$\overline{STROBE}$ will not generate an interrupt because the positive portion of $\overline{STROBE}$ is less than the width of $\overline{M1}$ and as such will not generate an interrupt due to the internal logic configuration of the PIO.

If the PIO is not in a reset status (i.e. a control mode has been selected), the output register may be loaded before Mode 0 is selected. This allows port output lines to become active in a user defined state. For example, assume the outputs are desired to become active in a logic one state, the following would be the initialization sequence:

a) PIO RESET
b) Load Interrupt Vector
c) Select Mode 1 (input) (automatic due ro RESET)
d) Write FF to Data Port
e) Select Mode 0 (Outputs go to "1's")
f) Enable Interrupt if desired

MODE 0 (OUTPUT) TIMING - READY TIED TO STROBE
Figure 5.0-1c

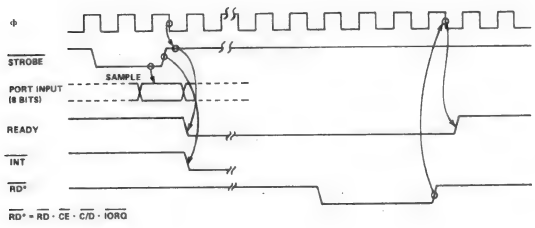


5.2 INPUT MODE (MODE 1)

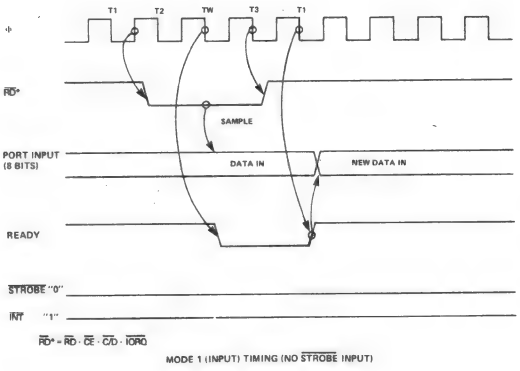
Figure 5.0-2 illustrates the timing of an input cycle. The peripheral initiates this cycle using The STROBE line after the CPU has performed a data read. A low level on this line loads data into the port input register and the rising edge of the STROBE line activates the interrupt request line (INT) if the interrupt enable is set and this is the highest priority requesting device. The next falling edge of the clock line (Φ) will then reset the READY line to an inactive state signifying that the input register is full and further loading must be inhibited until the CPU reads the data. The CPU will in the course of its interrupt service routine, read the data from the interrupting port. When this occurs, the positive edge from the CPU RD signal will raise the READY line with the next low going transition of Φ, indicating that new data can be loaded into the PIO.

Since RESET causes READY to go low a dummy Input instruction may be needed in some systems to cause READY to go high the first time in order to start "handshaking".

MODE 1 (INPUT) TIMING
Figure 5.0-2a



MODE 1 (INPUT) TIMING (NO STROBE INPUT)
Figure 5.0-2b



MODE 1 (INPUT) TIMING (NO STROBE INPUT)

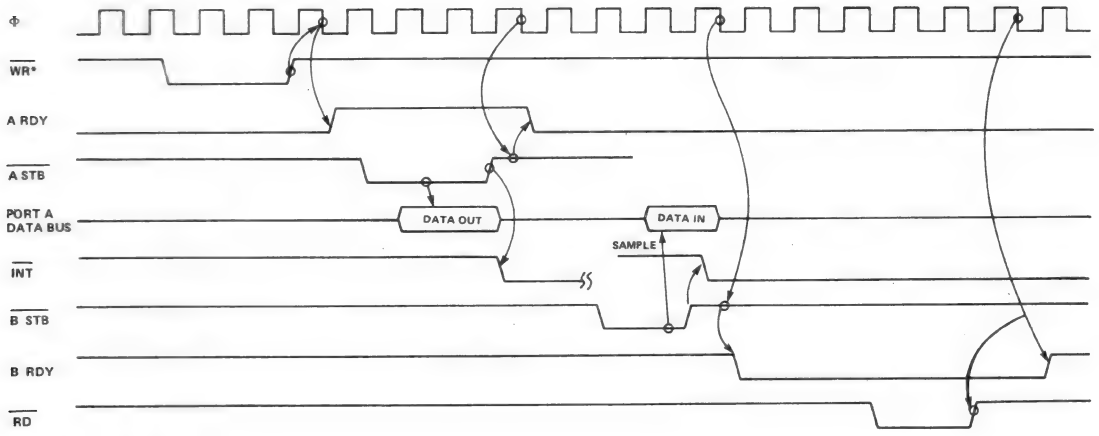
If already active, READY will be forced low one and one-half Φ periods following the falling edge of IORQ during a read of a PIO port as shown in Figure 5.0-2b. If the user strobes data into the PIO only when READY is high, the forced state of READY will prevent input register data from changing while the CPU is reading the PIO. Ready will go high again after the rising edge of the IORQ as previously described.

5.3 BIDIRECTIONAL MODE (MODE 2)

This mode is merely a combination of Mode 0 and Mode 1 using all four handshake lines. Since it requires all four lines, it is available only on Port A. When this mode is used on Port A, Port B must be set to the Bit Control Mode. The same interrupt vector will be returned for a Mode 3 interrupt on Port B and an input transfer interrupt during Mode 2 operation of Port A. Ambiguity is avoided if Port B is operated in a polled mode and the Port B mask register is set to inhibit all bits.

Figure 5.0-3 illustrates the timing for this mode. It is almost identical to that previously described for Mode 0 and Mode 1 with the Port A handshake lines used for output control and the Port B lines used for input control. The difference between the two modes is that, in Mode 2, data is allowed out onto the bus only when the A $\overline{\text{STROBE}}$ is low. The rising edge of this strobe can be used to latch the data into the peripheral since the data will remain stable until after this edge. The input portion of Mode 2 operates identically to Mode 1. Note that both Port A and Port B must have their interrupts enabled to achieve an interrupt driven bidirectional transfer.

PORT A, MODE 2 (BIDIRECTIONAL) TIMING
Figure 5.0-3



$\overline{WR}^* = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$
$\overline{RD}^* = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

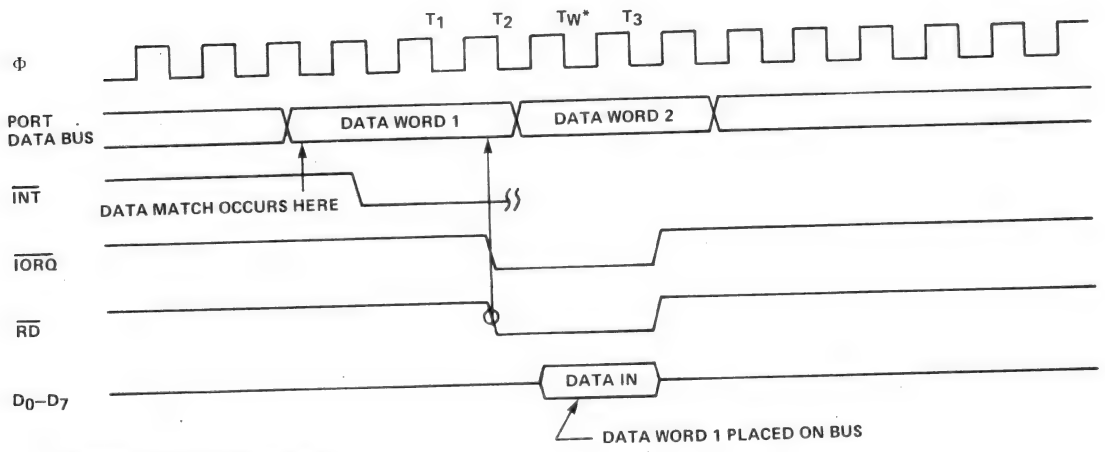
The peripheral must not gate data onto a port data bus while $\overline{\text{A STB}}$ is active. Bus contention is avoided if the peripheral uses $\overline{\text{B STB}}$ to gate input data onto the bus. The PIO uses the $\overline{\text{B STB}}$ low level to sample this data. The PIO has been designed with a zero hold time requirement for the data when latching in this mode so that this simple gating structure can be used by the peripheral. That is, the data can be disabled from the bus immediately after the strobe rising edge. Note that if $\overline{\text{A STB}}$ is low during a read operation of Port A (in response to a $\overline{\text{B STB}}$ interrupt) the data in the output register will be read by the CPU instead of the correct data in the data input register. The correct data is latched in the input register it just cannot be read by the CPU while $\overline{\text{A STB}}$ is low. If the $\overline{\text{A STB}}$ signal could go low during a CPU Read, it should be blocked from reaching the $\overline{\text{A STB}}$ input of the PIO while BRDY is low (the CPU read will occur while BRDY is low as the $\overline{\text{RD}}$ signal returns BRDY high).

5.4 CONTROL MODE (MODE 3)

The control mode does not utilize the handshake signals and a normal port write or port read can be executed at any time. When writing, the data will be latched into output registers with the same timing as Mode 0. A RDY will be forced low whenever Port A is operated in Mode 3. B RDY will be held low whenever Port B is operated in Mode 3 unless Port A is in Mode 2. In the latter case, the state of B RDY will not be affected.

When reading the PIO, the data returned to the CPU will be composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register will contain data which was present immediately prior to the falling edge of $\overline{RD}$. See Figure 5.0-4.

MODE 3 TIMING
Figure 5.0-4a



*Timing Diagram Refers to Bit Mode Read

An interrupt will be generated if interrupts from the port are enabled and the data on the port data lines satisfies the logical equation defined by the 8-bit mask control registers. Another interrupt will not be generated until a change occurs in the status of the logical equation. A Mode 3 interrupt will be generated only if the result of a Mode 3 logical operation changes from false to true. For example, assume that the Mode 3 logical equation is an "OR" function. An unmasked port data line becomes active and an interrupt is requested. If a second unmasked port data line becomes active concurrently with the first, a new interrupt will not be requested since a change in the result of the Mode 3 logical operation has not occurred. Note that port pins defined as outputs can contribute to the logical equation if their bit positions are unmasked.

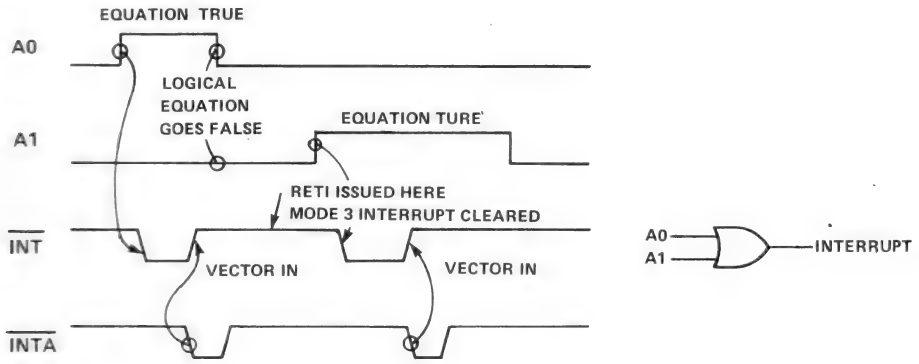
If the result of a logical operation becomes true immediately prior to or during $\overline{M1}$, an interrupt will be requested after the trailing edge of $\overline{M1}$, provided the logical equation remains true after $\overline{M1}$ returns high.

Figure 5.0-4b is an example of Mode 3 interrupts. The port has been placed in Mode 3 and OR logic selected and signals are defined to be high. All but bits A0 and A1 are masked out and are not monitored thereby creating a two input positive logic OR gate. In the timing diagram A0 is shown going high and creating an interrupt ($\overline{INT}$ goes low) and the CPU responds with an Interrupt Acknowledge cycle ($\overline{INTA}$). The PIO port with its interrupt pending sends in its Vector and the CPU goes off into the Interrupt Service Routine. A0 is shown going inactive either by itself or perhaps as a result of action taken in the Interrupt Service Routine (making the logical equation false). An arrow is shown at the point in time where the Service Routine issues the RETI instruction which clears the PIO interrupt structure. A1 is next shown going high making the logical equation-true and generating another interrupt. Two important points need to be made from this example:

1) A1 must not go high before A0 goes low or else the logical equation will not go false – a requirement for A1 to be able to generate an interrupt.

2) In order for A1 to generate an interrupt it must be high after the RETI issued by A0's Service Routine clears the PIO's Interrupt structure. In other words, if A1 were a positive pulse that occurred after A0 went low (to make the equation false) and went low before the RETI had cleared the Interrupt Structure it would have been missed. The logic equation must become false after the INTA for A0's service and then must be true or go true after RETI clears the previous interrupt for another interrupt to occur.

MODE 3 EXAMPLE
Figure 5.0-4b



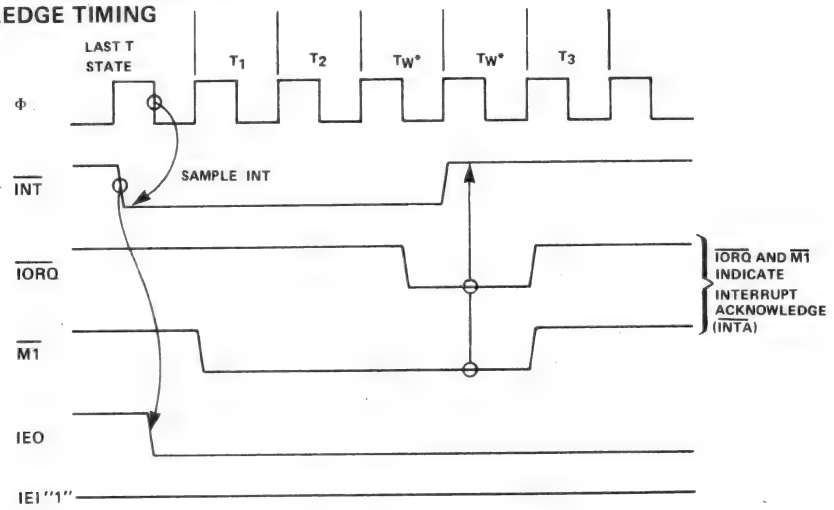
6.0 INTERRUPT SERVICING

Some time after an interrupt is requested by the PIO, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). During this time the interrupt logic of the PIO will determine the highest priority port which is requesting an interrupt. (This is simply the device with its Interrupt Enable Input high and its Interrupt Enable Output low). To insure that the daisy chain enable lines stabilize, devices are inhibited from changing their interrupt request status when $\overline{M1}$ is active. The highest priority device places the contents of its interrupt vector register onto the Z80 data bus during interrupt acknowledge.

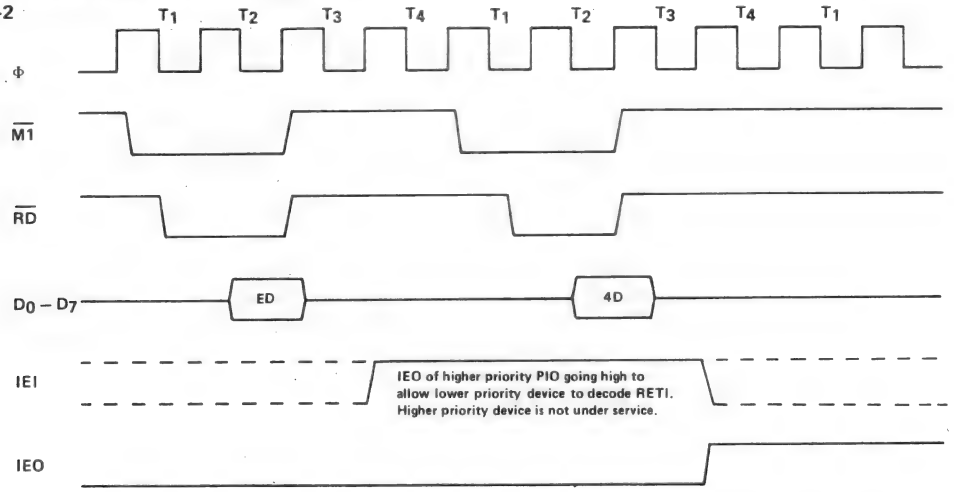
Figure 6.0-1 illustrates the timing associated with interrupt requests. During $\overline{M1}$ time, no new interrupt requests can be generated. This gives time for the Int Enable signals to ripple through up to four PIO circuits. The PIO with IEI high and IEO low during $\overline{INTA}$ will place the 8-bit interrupt vector of the appropriate port on the data bus at this time.

If an interrupt requested by the PIO is acknowledged, the requesting port is 'under service'. IEO of this port will remain low until a return from interrupt instruction (RETI) is executed while IEI of the port is high. If an interrupt request is not acknowledged, IEO will be forced high for one $\overline{M1}$ cycle after the PIO decodes the opcode 'ED'. This action guarantees that the two byte RETI instruction is decoded by the proper PIO port. See Figure 6.0-2.

INTERRUPT ACKNOWLEDGE TIMING
Figure 6.0-1



RETURN FROM INTERRUPT CYCLE
Figure 6.0-2



DAISY CHAIN INTERRUPT SERVICING
Figure 6.0-3

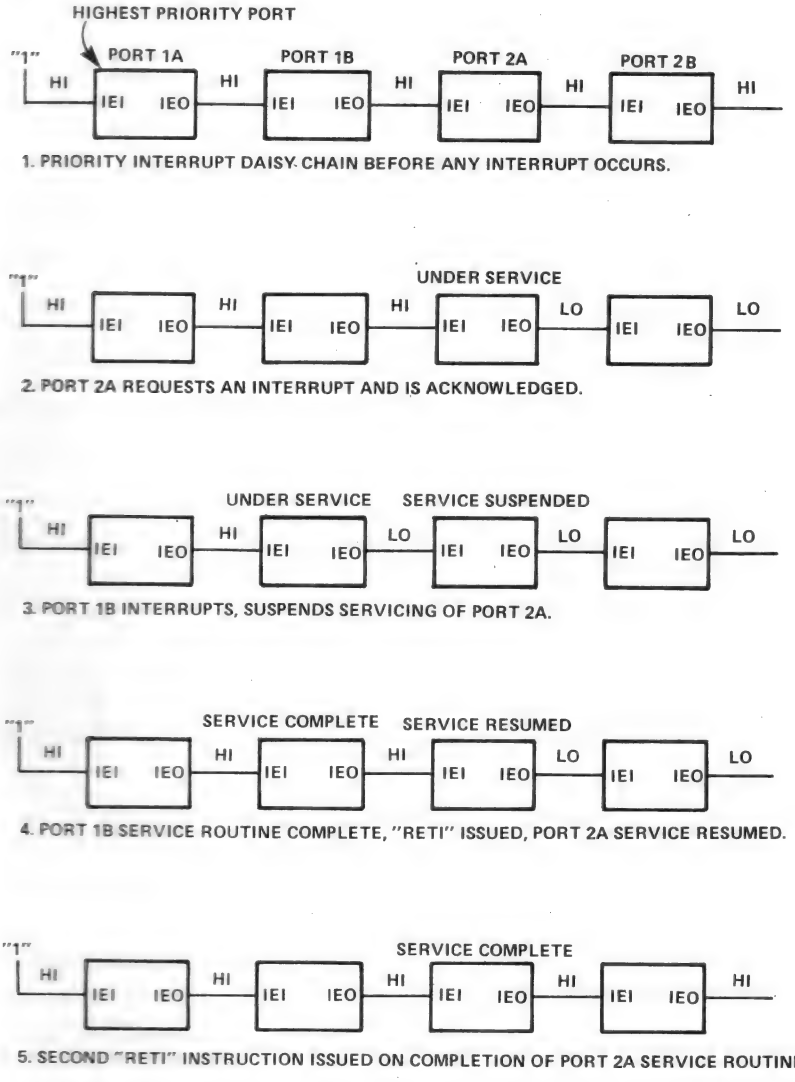


Figure 6.0-3 illustrates a typical nested interrupt sequence that could occur with four ports connected in the daisy chain. In this sequence Port 2A requests and is granted an interrupt. While this port is being serviced, a higher priority port (1B) requests and is granted an interrupt. The service routine for the higher priority port is completed and a RETI instruction is executed to indicate to the port that its routine is complete. At this time the service routine of the lower priority port is completed.

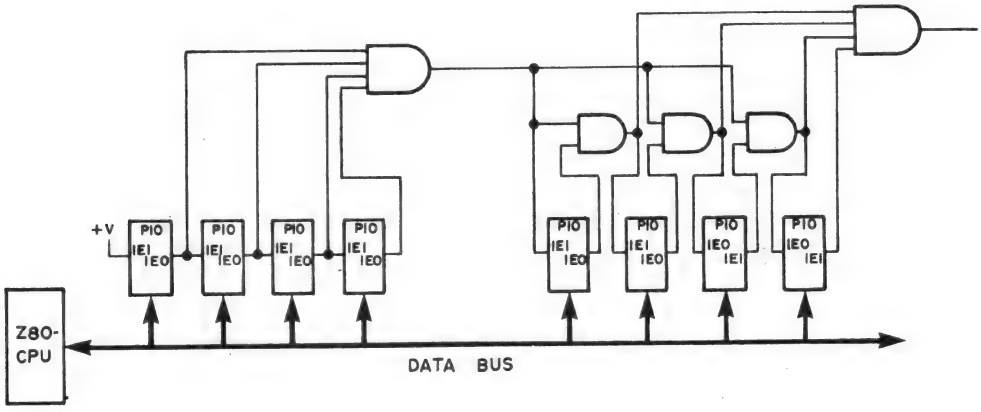
7.0 APPLICATIONS

7.1 EXTENDING THE INTERRUPT DAISY CHAIN

Without any external logic, a maximum of four Z80-PIO devices may be daisy chained into a priority interrupt structure. This limitation is required so that the interrupt enable status (IEO) ripples through the entire chain between the beginning of $\overline{M1}$, and the beginning of $\overline{IORQ}$ during an interrupt acknowledge cycle. Since the interrupt enable status cannot change during $\overline{M1}$, the vector address returned to the CPU is assured to be from the highest priority device which requested an interrupt.

If more than four PIO devices must be accommodated, a "look-ahead" structure may be used as shown in figure 7.0-1. With this technique more than thirty PIO's may be chained together using standard TTL logic.

A METHOD OF EXTENDING THE INTERRUPT PRIORITY DAISY CHAIN
Figure 7.0-1



7.2 I/O DEVICE INTERFACE

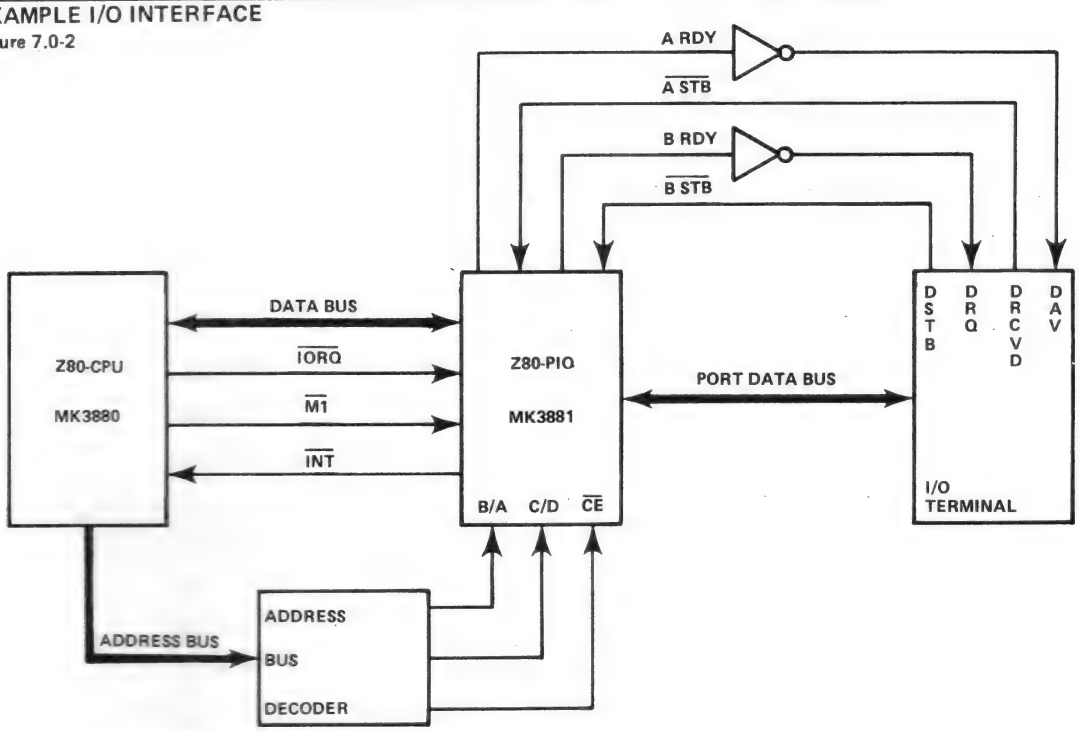
In this example, the Z80-PIO is connected to an I/O terminal device which communicates over an 8 bit parallel bidirectional data bus as illustrated in figure 7.0-2. Mode 2 operation (bidirectional) is selected by sending the following control word to Port A:

EXAMPLE I/O INTERFACE
Figure 7.0-2

D7	D6	D5	D4	D3	D2	D1	D0
1	0	X	X	1	1	1	1

MODE CONTROL

EXAMPLE I/O INTERFACE
Figure 7.0-2



Next, the proper interrupt vector is loaded (refer to CPU Manual for details on the operation of the interrupt).

V7	V6	V5	V4	V3	V2	V1	0

Interrupts are then enabled by the rising edge of the first $\overline{M1}$ after the interrupt mode word is set unless that $\overline{M1}$ defines an interrupt acknowledge cycle. If a mask follows the interrupt mode word, interrupts are enabled by the rising edge of the first $\overline{M1}$ following the setting of the mask.

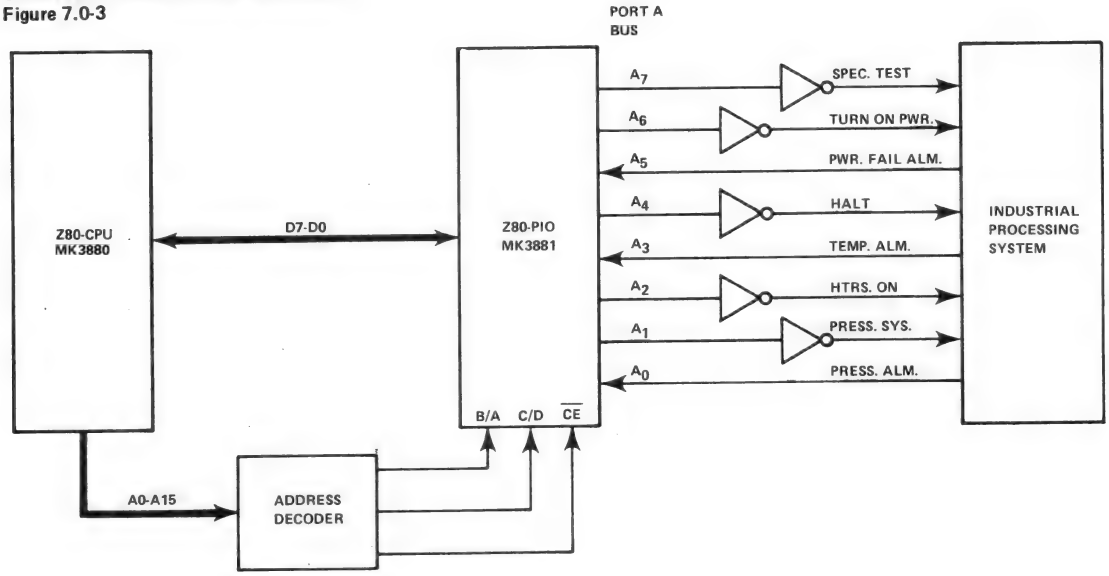
Data can now be transferred between the peripheral and the CPU. The timing for this transfer is as described in Section 5.0.

7.3 CONTROL INTERFACE

A typical control mode application is illustrated in figure 7.0-3. Suppose an industrial process is to be monitored. The occurrence of any abnormal operating condition is to be reported to a Z80-CPU based control system. The process control and status word has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Special Test	Turn On Power	Power Failure Alarm	Halt Process-ing	Temp. Alarm	Temp Heaters On	Pressur-ize System	Pressure Alarm

CONTROL MODE APPLICATION
Figure 7.0-3



The PIO may be used as follows. First Port A is set for Mode 3 operation by writing the following control word to Port A.

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	1	1	1	1

Whenever Mode 3 is selected, the next control word sent to the port must be an I/O select word. In this example we wish to select port data lines A5, A3, and A0 as inputs and so the following control word is written:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1	0	0	1

Next the desired interrupt vector must be loaded (refer to the CPU manual for details);

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

An interrupt control word is next sent to the port:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	0	1	1	1
Enable Interrupts	OR Logic	Active High	Mask Follows	Interrupt Control			

The mask word following the interrupt mode word is:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	1	0

Selects A5, A3 and A0 to be monitored

Now, if a sensor puts a high level on line A5, A3, or A0, an interrupt request will be generated. The mask word may select any combination of inputs or outputs to cause an interrupt. For example, if the mask word above had been:

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	1	0	1	1	0

then an interrupt request would also occur if bit A7 (special Test) of the output register was set.

Assume that the following port assignments are to be used:

$E0_H$= Port A Data
$E1_H$= Port B Data
$E2_H$= Port A Control
$E3_H$= Port B Control

All port numbers are in hexadecimal notation. This particular assignment of port numbers is convenient since A_0 of the address bus can be used as the Port B/A Select and A_1 of the address bus can be used as the Control/Data Select. The Chip Enable would be the decode of CPU address bits A_7 thru A_2 (111000). Note that if only a few peripheral devices are being used, a Chip Enable decode may not be required since a higher order address bit could be used directly.

8.0 PROGRAMMING SUMMARY

8.1 LOAD INTERRUPT VECTOR

V7	V6	V5	V4	V3	V2	V1	0

8.2 SET MODE

M1	M0	X	X	1	1	1	1

MODE NUMBER	M_1	M_0	MODE
0	0	0	Output
1	0	1	Input
2	1	0	Bidirectional
3	1	1	Bit Control

When selecting Mode 3, the next word to the PIO must set the I/O Register:

I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O = 1 Sets bit to Input
I/O = 0 Sets bit to Output

8.3 SET INTERRUPT CONTROL

Int Enable	AND/ OR	High/ Low	Mask Follows	0	1	1	1

USED IN MODE 3 ONLY

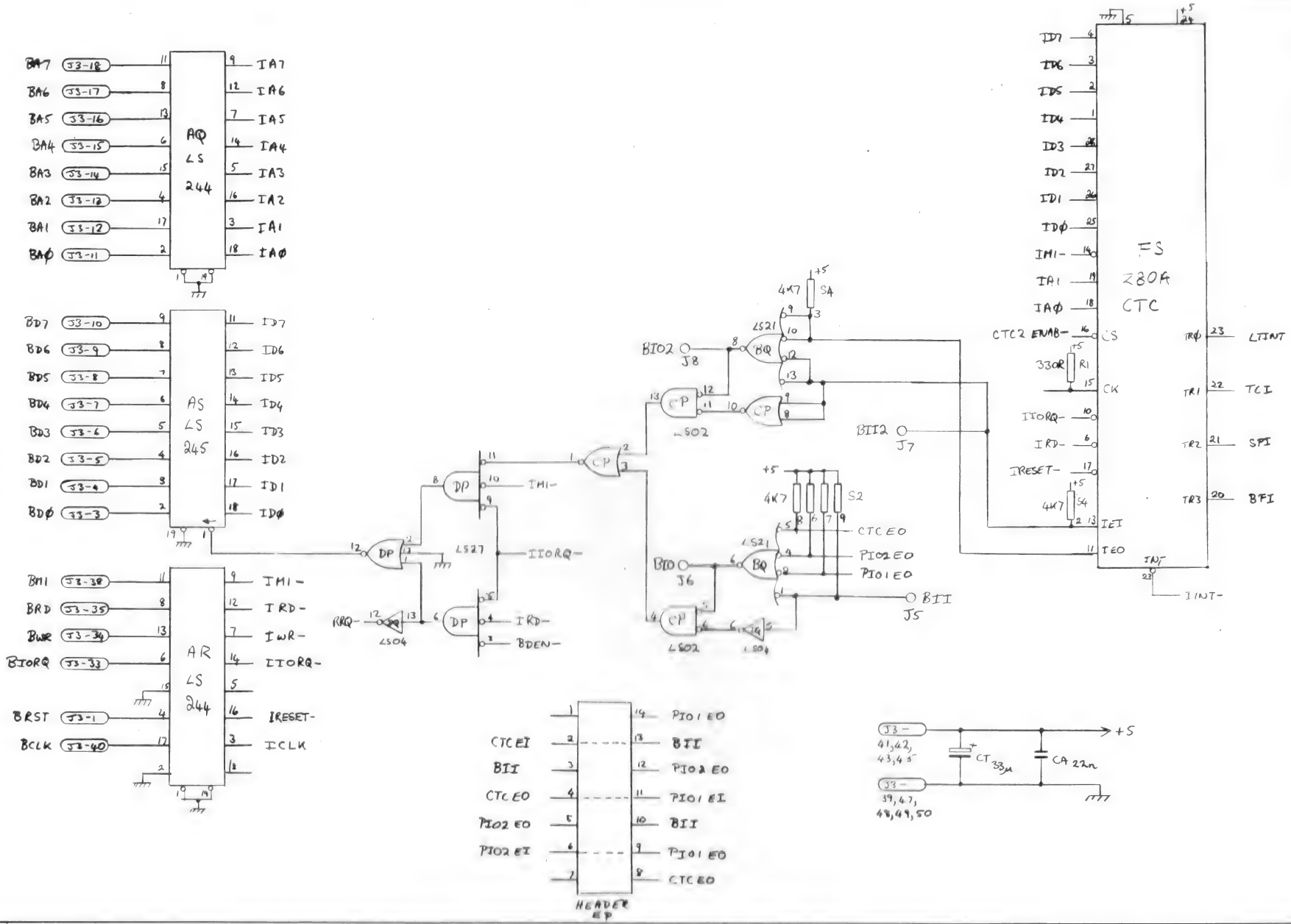
If the "mask follows" bit is high, the next control word written to the PIO must be the mask:

MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

MB = 0, Monitor bit
MB = 1, Mask bit from being monitored

Also, the interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1



B.20

MODIFICATION	
A	10/11/80
B	10/6/80

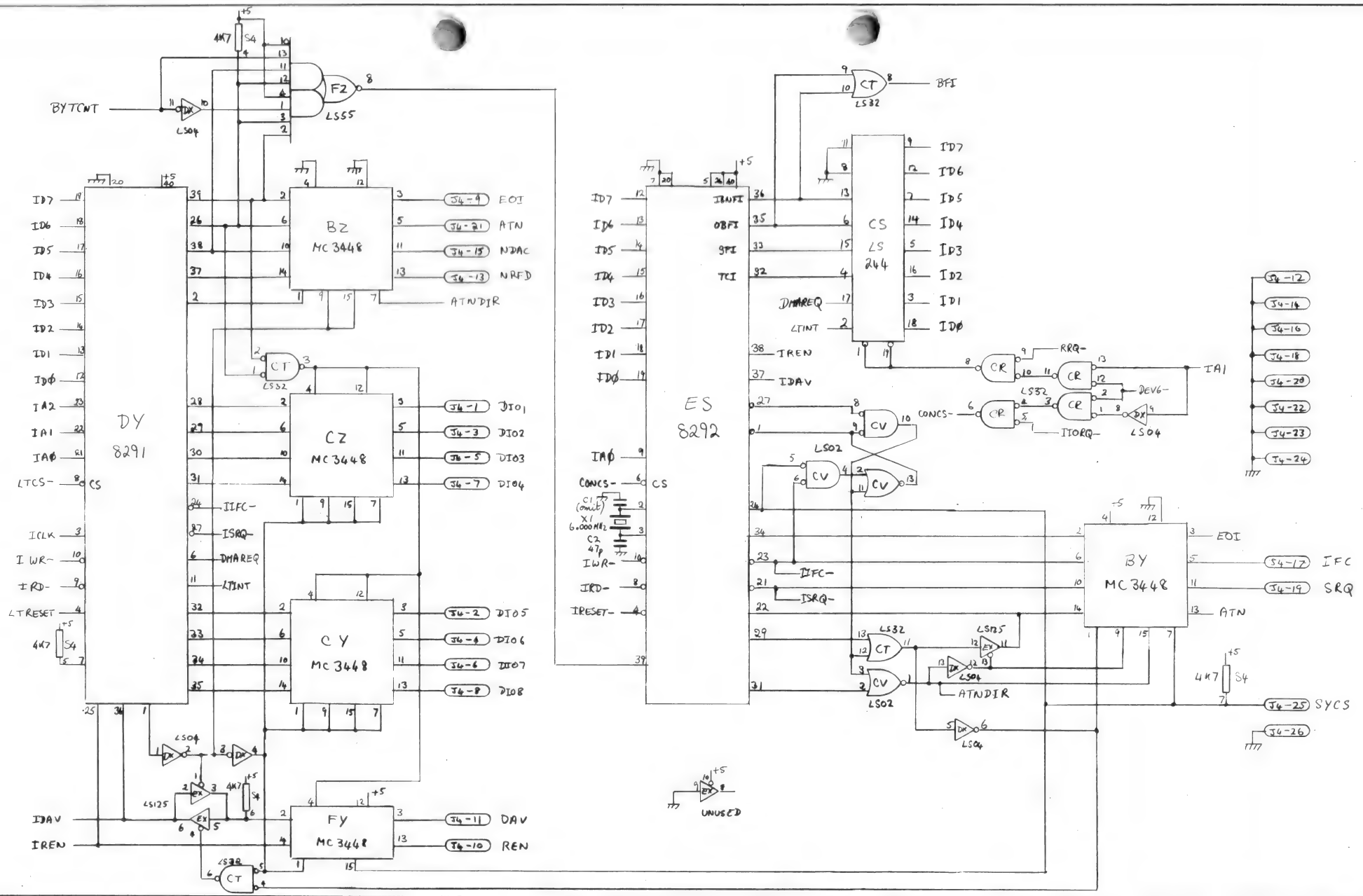
WP 1001

Research Machines Ltd

DATE	DRAWN BY	CHECKED BY
10/6/80	T.B. Baker	

TITLE: IEEE-488 ~ PIO INTERFACE

DRAWING NO. 380Z-IEEE-1



B.21

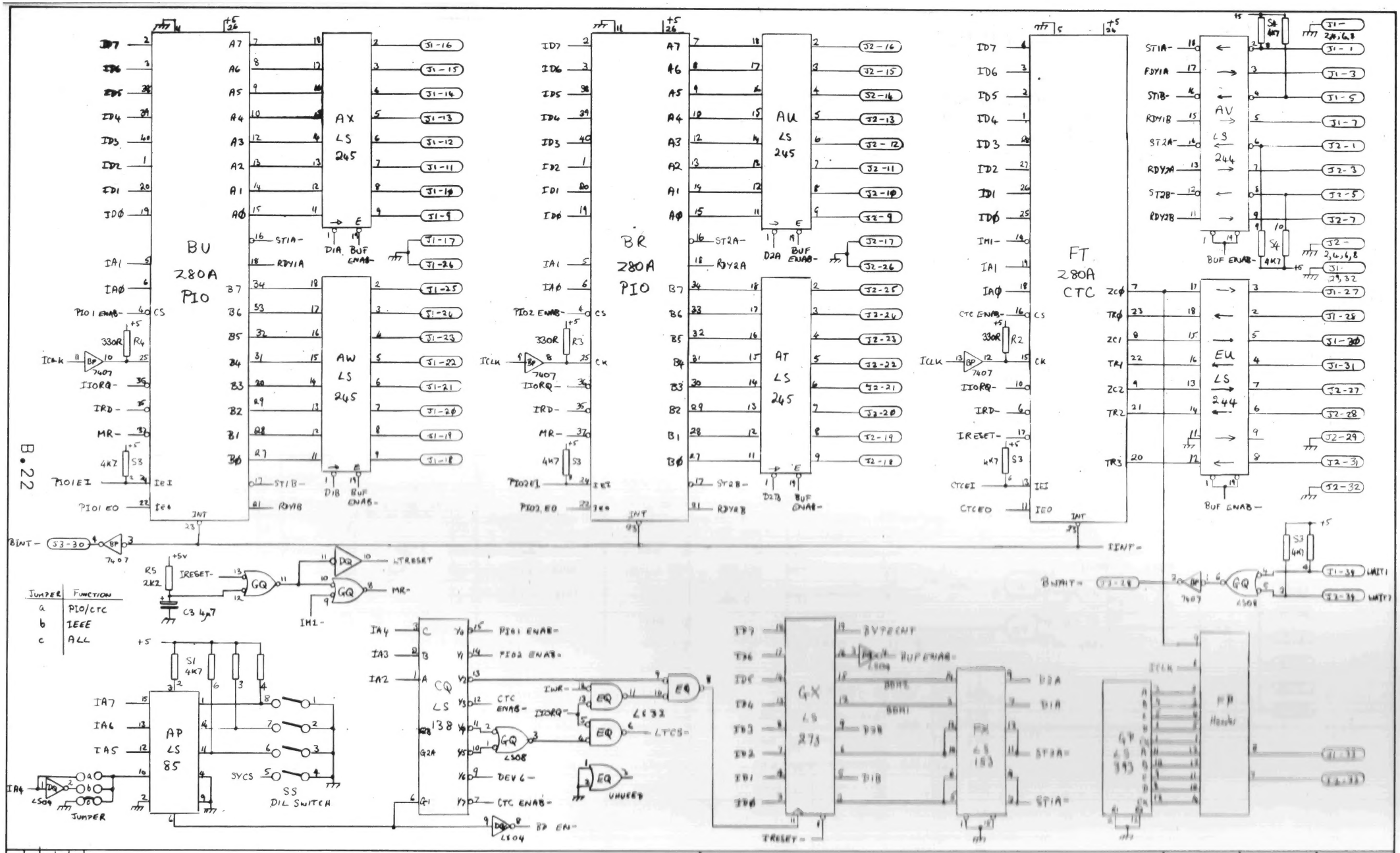
MODIFICATION	
A	10/11/80
B	10/6/80
C	29/9/80
D	17/9/80

Research Machines Ltd

DATE	DRAWN BY	CHECKED BY
10/6/80	T.B.	

TITLE: IEEE-488 ~ PIO INTERFACE

DRAWING NO.: 380Z-IEEE-2



B.22

Jumper	Function
a	PIO/CTC
b	IEEE
c	ALL

Modification		
A	10/1/80	TBR
B	9/6/80	TBL
C	21/7/80	TBL
D	20-2-80	GAK

Research Machines Ltd

Date	Drawn by	Checked by
9/6/80	TBL	

TITLE: IEEE-488 PIO INTERFACE

DRAWING NO. 380Z-IEEE-3

WP1001

INDEX

IPN 01138101